XYZZY

Link submit:

https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=1498

Solution:

C++: http://ideone.com/r7N4KP

python: https://ideone.com/l9efbC (chỉ để tham khảo vì vùng nhớ python sẽ bị tràn và bị RE)

Java: https://ideone.com/GOrvZ8

Tóm tắt đề: Cho một trò chơi với N phòng, mỗi phòng có một mức năng lượng có giá trị trong đoạn [-100, 100], một người chơi ban đầu có mức năng lượng là 100, khi đi qua 1 phòng thì sẽ nhận thê được mức năng lượng tại phòng đó, hỏi liệu có tồn tại cách đi sao cho có thể di chuyển từ phòng 1 đến phòng n mà không bị hết năng lượng hay không. Người chơi có thể đi qua một phòng nhiều lần và mỗi lần đi qua đề nhận được năng lượng tại phòng đó.

Input

Gồm nhiều test case, mỗi test case bắt đầu bằng 1 dòng chứa 1 số nguyên n là số lượng phòng ($1 \le n \le 100$).

N dòng tiếp theo, dòng thứ i bắt đầu bằng một số là năng lượng tại phòng i, tiếp theo là một số m là số lượng phòng mà có thể đi đến được từ phòng i, m số cuối cùng là danh sách các phòng đến được từ phòng i.

Input kết thúc ghi gặp n = -1.

Output

Với mỗi test case in ra trên 1 dòng, nếu có cách đi đến phòng n thì in ra “‘winnable”, ngược lại thì in ra “‘hopeless”.

5 0 1 2 -60 1 3 -60 1 4 20 1 5 0 0 5	hopeless hopeless winnable winnable

0 1 2 20 1 3 -60 1 4 -60 1 5 0 0 5 0 1 2 21 1 3 -60 1 4 -60 1 5 0 0 5 0 1 2 20 2 1 3 -60 1 4 -60 1 5 0 0 -1	

Giải thích ví dụ:

Ví dụ 1: Đồ thị gồm các cạnh có hướng (1, 2), (2, 3), (3, 4), (4, 5) vì vậy chỉ có 1 cách đi từ 1 đến 5 là 1-2-3-4-5. Thì trên đường đi này, đi từ 1 qua 2, năng lượng là 100. Đi từ 2 qua 3, năng lượng giảm 60, còn lại 40. Khi đi từ 3 qua 4, năng lượng giảm đi 60 tức là -20, lúc này nó đã không dương nên không thể chơi tiếp. Không thể đi đến được n nên kết quả là “hopeless”

Hướng dẫn giải:

Trong trường hợp đơn giản: nếu ta tìm được một đường đi từ 1 đến n mà có mức năng lượng dương thì chắc chắn kết quả là winnable. Trong trường hợp ngược lại, tức không có đường đi đơn nào mà có thể đi từ 1 đến n với mức năng lượng dương, thì lúc này ta vẫn chưa thể kết luận là không thể thắng. Vì có thể trên đường đi tồn tại một chu trình mà khi đi qua nó, năng lượng thu được sẽ tăng lên, thì ta đi càng nhiều trong chu trình thì năng lượng tăng lên càng cao, đến một lúc nào đó sẽ đủ lớn để có thể đi đến n. Như vậy, bài này ta có thể sử dụng Bellman Ford, trước tiên tìm đường đi “dài nhất” từ 1 đến tất cả các đỉnh còn lại. Sau đó thì:

- Nếu dist[n] > 0 thì hiển nhiên kết quả là winnable.
- Ngược lại, ta duyệt qua từng cạnh của đồ thị, nếu cạnh (u, v) thuộc một chu trình dương nào đó, thì lúc này ta biết chắc chắn rằng có cách đi từ 1 đến u và v, đồng thời có thể đi nhiều lần trong chu trình chứa cạnh (u, v). Nếu tồn tại một đường đi bất kì không cần quan tâm chi phí từ u đến n, thì chắc chắn sẽ có cách đi để thắng. Vậy thì ta sử dụng BFS/DFS để tìm đường đi từ u đến n, nếu tồn tại thì kết quả là winnable.

Độ phức tạp: Độ phức tạp thuật toán Bellman Ford là O(V*E). Với mỗi cạnh của đồ thị, ta có thể duyệt BFS lại một lần để kiểm tra chu trình (trường hợp mọi cạnh đều thuộc một chu trình âm nào đó), thì độ phức tạp ở phần kiểm tra là O(V*E). Vậy độ phức tạp chung của cả chương trình là O(V*E).